TECHNICAL REPORT

No. 3

A MATRIX-VECTOR FORMULATION OF THE QD DIGITAL FILTER

DECEMBER
1968

W. A. McCOOL

ANALYSIS AND COMPUTATION DIRECTORATE

WHITE SANDS MISSILE RANGE
NEW MEXICO

A Matrix-Vector Formulation of the QD Digital Filter
December 1968

Prepared by W. A. Mc COOL
Acting Director, Analysis & Computation Directorate

Reviewed by Mrs. Virginia Tipton
Mathematician, Math Services Division

Abstract

This report describes the development of an alternate formulation of the QD digital filter using matrix-vector notation and methods. The objectives of the report are to clarify the concepts on which the filter is based and to briefly indicate the structural similarity of the QD and Kalman filters as well as the difference between them. The technique for inserting into QD observed derivative data obtained independently of the normal input observations is briefly described.

TABLE OF CONTENTS

A Matrix-Vector Formulation of the QD Digital Filter

Introduction

In July 1967 the writer developed a new, more general formulation of a very efficient type of real-time digital filter and called it the "QD Filter".[1] Actually, the structures of the computational formulas for the first and second order QD filters were not new. As early as 1957,[2][3] for example, workers in radar engineering developed the well-known α-β digital filter which is used primarily to predict estimates of position coordinates. The α-β filter is comparable to a first-order QD type of filter. Then in 1964 J. J. Lynn [4] developed a second-order filter with three coefficients α,β, and γ analytically determined to attain the optimum trade-off between distortion of the true input information and attenuation of the corrupting noise inherent in the input data. The QD formulation, in contrast to these earlier developments, is much more general because it is based on an N-th order polynominal fit of a span of input data points, subject to the least-squares error criterion and intercept and slope constraints. Exhaustive experiments have shown that the performances of the second and third order QD filters are essentially the same as the respective performances of the equivalent classical constrained least-squares filters.

The minimal computing time required by the QD filter is independent of the point span length. Because of this computational efficiency, the second-order QD filter is now implemented in a variety of computer programs for filtering radar data at White Sands Missile Range, e.g., the real-time program for flight-safety support of PERSHING missile overflights. The third-order version has been used experimentally and displays negligible "velocity lag" error in velocity estimates obtained from radars observing a missile during its thrusting periods. It is understood that Lynn's α,β,γ digital filter has been used in similiar real-time computer programs at the Eastern Test Range.

It has been observed that the QD computational formulas have the same general structure as those of the discrete Kalman Filter. This similarity is quite natural because both filters are recursive. Otherwise, they are so different that the QD filter can not be considered as a special case of the Kalman filter, which will be quite apparent to readers who are familiar with the Kalman theory.

The original QD filter development employed classical summation (scalar) notation which was extremely tedious for the reader to follow. It is hoped that this new development using matrix-vector notation presented in the following chapters will be clearer and facilitate much easier reading. The manner in which the development is presented has several other specific objectives: (1) to show the Kalman-knowledgable readers that the QD filter can be cast in matrix-vector notation and that, despite the structural similarity, the QD theory is distinctly

different from the Kalman theory; and (2) to provide a self-contained QD filter development which can be understood without reference 1. The material presented in the last chapter on the extension and applications of the QD filter was not included in reference 1 but nevertheless is most significant.

Notation

In our development we will use the following notation:

1 - Matrices in capital letters or matrix algebraic expressions in brackets;

2 - Column vectors in lower-case letters;

3 - Row vectors in lower case letters with a "T" superscript;

4 - Scalars, vector elements, matrix elements in lower case Greek letters;

5 - Minus one superscript for the inverse of a square matrix;

6 - Subscripts on derivative vectors denote the point in the span at which the derivatives are evaluated;

7 - Subscripts will also be used to designate the components of partitioned matrices and vectors;

8 - Estimates will be designated with a "hat" and predicted values with a "bar"; and

9 - Superscripts in parentheses on derivative vectors' elements denote the order of the derivative.

10 - Formula numbers with an asterisk are rewritten in Appendix I along with the corresponding formulas of reference 1 whose numbers are given in brackets.

The Matrix-Vector Formulation of the Classical Constrained Least-Squares Curve Fitting Procedure

We will be fitting an N-th order polynomial to a span of (M+1) data points x_m, $0 \leq m \leq M$, $M \geq N$, equally spaced with an interval h. The Mth position of the span will contain the current (real-time) data point. For clearer delineation we will use the notation of a truncated Taylor series expanded about the point m=0 instead of a polynomial with undetermined coefficients. Accordingly, we write:

$$\hat{x}_m = \hat{x}_0 + mh\hat{\dot{x}}_0 + \frac{(mh)^2}{2}\hat{\ddot{x}}_0 + \ldots + \frac{(mh)^N}{N!}\hat{x}_0^{(N)}, \quad 0 \leq m \leq M, \tag{1}*$$

where:

$\hat{x}_m$ - Smoothed estimates of the x_m, the M+1 values of input data contained within the span, $0 \leq m \leq M$;

$\hat{x}_0^{(n)}$ - Estimates of the derivatives (parameters), $0 \leq n \leq N$, evaluated at m=0.

Given the x_m we wish to determine the values of the $\hat{x}_0^{(n)}$ which will minimize the squared-error sum ψ :

$$\psi = \sum_{m=0}^{M} \varepsilon_m^2 = \sum_{m=0}^{M} (x_m - \hat{x}_m)^2 \tag{2}*$$

In matrix-vector notation (1) and (2) are written as:

$$x - e = \hat{x} = A\hat{x}_0 \tag{3}*$$

and

$$\psi = e^T e = (x-\hat{x})^T(x-\hat{x}) \tag{4}*$$

where:

$$x \equiv \begin{vmatrix} x_0 \\ x_1 \\ \vdots \\ x_M \end{vmatrix}, \quad (5) \qquad \hat{x} \equiv \begin{vmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \vdots \\ \hat{x}_M \end{vmatrix}, \quad (6)$$

$$\hat{x}_0 \equiv \begin{vmatrix} \hat{x}_0 \\ \hat{x}_0 \\ \vdots \\ \hat{x}_0^{(N)} \end{vmatrix}, \quad (7) \qquad e \equiv \begin{vmatrix} x_0 - \hat{x}_0 \\ x_1 - \hat{x}_1 \\ \vdots \\ x_M - \hat{x}_M \end{vmatrix}, \quad (8)$$

$$A \equiv \begin{vmatrix} 1 & 0 & 0 & & 0 \\ 1 & h & \frac{h^2}{2} & \cdots & \frac{h^N}{N!} \\ 1 & 2h & \frac{(2h)^2}{2} & & \frac{(2h)^N}{N!} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & Mh & \frac{(Mh)^2}{2} & \cdots & \frac{(Mh)^N}{N!} \end{vmatrix}, \quad (9)$$

The conditions for which ψ is a minimum are determined by the methods of matrix calculus as follows:

$$\left.\frac{\partial \psi}{\partial x}\right|_{0 x_0 = \hat{x}_0} = -2(x-\hat{x})^T A = -2A^T(x-\hat{x}) = 0. \qquad (10)^*$$

Rearranging (10), substituting (3), and solving for the parameter estimates $\hat{x}_0$ gives

$$A^T x = A^T A \hat{x}_0,$$

or

$$\hat{x}_0 = [A^T A]^{-1} A^T x, \qquad (11)^*$$

Substituting (11) into (3) we obtain the desired estimates $\hat{x}$ with the matrix-vector formula for the classical unweighted least-squares curve fitting procedure, i.e.,

$$\hat{x} = A[A^T A]^{-1} A^T x. \qquad (12)^*$$

Now suppose we have the very practical situation that we require not only some estimate $\hat{x}_m$, $m \neq 0$, within the span, e.g., the mid-point, but also the derivative estimates $\hat{x}_m^{(n)}$, $1 \leq n \leq N$. $\hat{x}_m$ is given by the Taylor series (1) after the parameter estimates $\hat{x}_0$ are obtained with (11). The derivative estimates are obtained directly by successive differentiation of (1), i.e.,

$$\begin{aligned}
\hat{x}_m &= \hat{x}_0 + mh\dot{\hat{x}}_0 + \frac{(mh)^2}{2}\ddot{\hat{x}}_0 + \cdots + \frac{(mh)^N}{N!}\hat{x}_0^{(N)} \\
\dot{\hat{x}}_m &= \dot{\hat{x}}_0 + mh\ddot{\hat{x}}_0 + \cdots + \frac{(mh)^{N-1}}{(N-1)!}\hat{x}_0^{(N)} \qquad (13)^* \\
&\vdots \\
\hat{x}_m^{(N)} &= \hat{x}_0^{(N)}
\end{aligned}$$

In matrix-vector notation the system (13) becomes

$$\hat{x}_m = \Phi^m \hat{x}_0, \qquad (14)^*$$

where

$$\hat{x}_m = \begin{vmatrix} x_m \\ \dot{x}_m \\ \vdots \\ x_m^{(N)} \end{vmatrix}, \quad (15) \qquad \hat{x}_0 = \begin{vmatrix} x_0 \\ \dot{x}_0 \\ \vdots \\ x_0^{(N)} \end{vmatrix}, \quad (7)$$

$$\phi^m = \begin{vmatrix} 1 & mh & \frac{(mh)^2}{2} & \cdots & \frac{(mh)^N}{N!} \\ 0 & 1 & mh & \cdots & \frac{(mh)^{N-1}}{(N-1)!} \\ 0 & 0 & 1 & \cdots & \vdots \\ \vdots & & & & mh \\ 0 & 0 & \cdots & \cdots & 0 \quad 1 \end{vmatrix} \qquad (16)$$

The matrix ϕ is an example of the familiar transition matrix in [illegible] theory. The matrix ϕ has some very interesting properties:

a. The value of its determinant is unity.

b. It obeys the exponentiation law.

$$\phi^M \cdot \phi^N = \phi^{M+N}$$

c. Inversion is a special case of b,

$$\phi \cdot \phi^{-1} = I$$

d. It is an integration operator.

$$\hat{x}_1 = \hat{\Phi x}_0$$

Substituting (11) into (14) gives the formula for evaluating the required estimates,

$$\hat{x}_m = \Phi^m [A^T A]^{-1} A^T x. \qquad (17)*$$

The foregoing development has been presented to demonstrate our matrix-vector notation in obtaining the familiar classical least-squares curve fitting procedure. The constrained procedure begins with the mathematical specification of the "intercept" and "slope" constraints. The intercept parameter is $\hat{x}_0$ and the slope parameter is $\hat{\dot{x}}_0$. When the constraints are not applied, $\hat{x}_0$ and $\hat{\dot{x}}_0$ are evaluated along with the remaining parameters using (11). When the constraints are applied, $\hat{x}_0$ and $\hat{\dot{x}}_0$ are obtained using (14) with m=1, i.e.,

$$\bar{\bar{x}}_0 = \Phi \hat{\bar{x}}_0 , \qquad (18)$$

where:

$$\bar{\bar{x}}_0 \equiv \begin{vmatrix} \bar{x}_0 \\ \bar{\dot{x}}_0 \\ \cdot \\ \cdot \\ \bar{x}_0^{(N)} \end{vmatrix}, \quad (19) \qquad \hat{\bar{x}}_0 \equiv \begin{vmatrix} \hat{x}_0 \\ \hat{\dot{x}}_0 \\ \cdot \\ \cdot \\ \hat{x}_0^{(N)} \end{vmatrix}, \quad (20)$$

The $\hat{\bar{x}}_0$ are the parameter estimates obtained in the preceding computation step. The $\bar{\bar{x}}_0$ are the predicted values of the current parameter estimates which are precisely the $\hat{x}_1$ obtained in the preceding computation step. We apply the intercept and slope constraints by imposing the conditions:

$$\hat{x}_0 = \bar{x}_0, \quad \text{and} \quad \hat{\dot{x}}_0 = \bar{\dot{x}}_0. \tag{21}$$

Application of these particular constraints is justified by physical considerations. If, for example, $\hat{x}_0$ and $\hat{\dot{x}}_0$ are, respectively, estimates of observed position and velocity components of a missile in flight, the known maximum acceleration must limit abrupt changes in position and velocity. Thus, the change of intercept and slope from one step to the next is specified in the transition matrix Φ in (18).

We will now evaluate the remaining parameter estimates under the constraint conditions (21). The first step is to partition A and $\hat{x}_0$ in (3)

$$\hat{x} = A\hat{x}_0 = A_1(\hat{x}_0)_1 + A_2(\hat{x}_0)_2 \tag{22}$$

where

$$\hat{x}_0 \equiv \begin{vmatrix} (\hat{x}_0)_1 \\ (\hat{x}_0)^2 \end{vmatrix} = \begin{vmatrix} \hat{x}_0 \\ \hat{\dot{x}}_0 \\ - - - \\ \hat{\ddot{x}}_0 \\ \vdots \\ \hat{x}_0^{(N)} \end{vmatrix}, \tag{23}$$

$$A \equiv [A_1 \; A_2] = \left| \begin{array}{cc|ccccc} 1 & 0 & 0 & 0 & . & . & 0 \\ 1 & h & \frac{h^2}{2} & & . & . & \frac{h^N}{N!} \\ 1 & 2h & \frac{(2h)^2}{2} & & . & . & \frac{(2h)^N}{N!} \\ \vdots & \vdots & & & & & \\ 1 & Mh & . & . & . & . & \frac{(Mh)^N}{N!} \end{array} \right| \tag{24}$$

From (22) we specify a new system:

$$\hat{x}_c = \hat{x} - A_1(\hat{x}_0)_1 = A_2(\hat{x}_0)_2 \tag{25}$$

We observe that

$$(x-\hat{x}) = (x - A_1(\hat{x}_0)_1 - \hat{x} + A_1(\hat{x}_0)_1) = (x_c - \hat{x}_c) \tag{26}$$

so that the sum of the squared errors can be written

$$\psi = (x_c - \hat{x}_c)^T (x_c - \hat{x}_c). \tag{27}$$

Then

$$\frac{\partial \psi}{\partial (\hat{x}_0)_2} = -2(x_c - \hat{x}_c)^T A_2 = -2A_2^T (x_c - \hat{x}_c) = 0. \tag{28}*$$

Substituting (25) into (28) and solving for $(\hat{x}_0)_2$ gives

$$(\hat{x}_0)_2 = [A_2^T A_2]^{-1} A_2^T x_c. \tag{29}*$$

Going back to (14) we can evaluate derivative estimates at any point within the span:

$$\hat{x}_m = \Phi^m \hat{x}_0 = \Phi^m \begin{vmatrix} (\hat{x}_0)_1 \\ (\hat{x}_0)_2 \end{vmatrix} = \Phi^m \begin{vmatrix} \hat{x}_0 = \bar{x}_0 \\ \hat{\dot{x}}_0 = \bar{\dot{x}}_0 \\ [A_2^T A_2]^{-1} A_2^T x_c \end{vmatrix} \tag{30}*$$

In this constrained least-squares formulation the $(\hat{x}_0)_1$ for the <u>next</u> step are evaluated using (30) with m=1, which implicitly imposes the conditions (21). Also, with m=M, (30) provides the real-time derivative estimates.

It is important to note here that this constrained least-squares curve-fitting procedure is inherently recursive because of the very nature of the intercept and slope constraints, i.e.,using information obtained in the preceding step to evaluate the estimates in the current step. It follows, then, that the procedure must be "initialized" with initial constraint values which are consistent with subsequent data. Any appreciable inconsistency generates an undesirable transient which will mask out the desired estimates until the transient "settles out". In practice, the initial constraints are obtained with an unconstrained curve fit over the first M+1 data points.

Matrix - Vector Formulation of the QD Filter

The key to this development of the QD filter is an approximation requiring reformulation of the conditions(28) which minimize the sum of the squared errors under the slope and intercept constraints. From (28) we have

$$A_2^T x_c = A_2^T A_2 (\hat{x}_0)_2. \tag{31}*$$

From (25) and (26) we can write

$$x_c = x - A_1(\hat{x}_0)_1 - A_2(\bar{x}_0)_2 + A_2(\bar{x}_0)_2 = \delta x + A_2(\bar{x}_0)_2, \tag{32}$$

where:

$$\delta x \equiv x - A_1(\hat{x}_0)_1 - A_2(\bar{x}_0)_2. \tag{33}*$$

Substituting (32) into (31) and rearranging gives

$$A_2^T \delta x = A_2^T A_2 \delta\hat{x}_0, \tag{34}*$$

where:

$$\delta\hat{x}_0 \equiv (\hat{x}_0)_2 - (\bar{x}_0)_2, \tag{35}*$$

or

$$(\hat{x}_0)_2 = (\bar{x}_0)_2 + \delta\hat{x}_0. \tag{36}$$

Solving (34) for the $\delta\hat{x}_0$ gives

$$\delta\hat{x}_0 = [A_2^T A_2]^{-1} A_2^T \delta x. \qquad (37)^*$$

Substituting (37) into (36) gives

$$(\hat{x}_0)_2 = (\bar{x}_0)_2 + [A_2^T A_2]^{-1} A_2^T \delta x, \qquad (38)^*$$

in which the unconstrained parameter estimates $(\hat{x}_0)_2$ are computed as sums of the predicted values and corrections. Clearly, (38) has no computational advantage over (29) but was developed to provide a basis for the QD approximation.

We observe in (33) that the δx vector contains the last (M+1) input data points, whereas the QD filter uses only the current data point. If (38) were used to evaluate the unconstrained parameter estimates, an x array would have to be retained to evaluate the δx in each computing step. An alternative procedure would be to retain a δx array which would have to be shifted in each computing step as well as corrected to take into account the new values of parameter estimates. With this arrangement each input data point makes a contribution through δx to the evaluation of the parameter estimates. Moreover, each data point makes a contribution for (M+1) consecutive computing steps as it "passes through the filter". The QD alternative is to let each data point make all of its (M+1) contributions during the current computing step and drop the contributions of the other M data points. To implement this reasoning we set each δx element equal to the current element δx_M minus the sum of the corrections it would receive as the filter processes the current and subsequent M data points. For example, the mth element of δx for a second-order constrained filter would be

$$\delta x_m = \delta x_M - \left| \frac{(Mh)^2}{2}\delta\hat{\ddot{x}}_{0,M} + \frac{(M-1)^2h^2}{2}\delta\hat{\ddot{x}}_{0,M-1} + \cdots + \frac{(M-m+1)^2h^2}{2}\delta\hat{\ddot{x}}_{0,M-m+1} \right|, \qquad 1 \le m \le M \qquad (39)$$

in which the $\delta\hat{\ddot{x}}_{0,k}$ is the correction for the unconstrained parameter estimate used to compute the correction for δx_m at the kth computing step after the corresponding input data point entered the filter. Since each $\delta\hat{\ddot{x}}_{0,k}$ in (39) can not be determined until the next M data points are processed, we will assume their average value is $\delta\hat{\ddot{x}}_0$, the parameter correction vector which is being evaluated during the current computing step. Applying this assumption, (39) can be written in the matrix-vector form:

$$\delta x = v\delta x_M - C^{(2)} v\delta\hat{x}_0, \tag{40}$$

where:

$$C^{(2)} = \begin{vmatrix} \frac{h^2}{2} & \frac{(2h)^2}{2} & \cdots & \frac{(Mh)^2}{2} \\ \frac{(2h)^2}{2} & \frac{(3h)^2}{2} & & \\ \vdots & \vdots & & \\ \vdots & \frac{(Mh)^2}{2} & & \\ \frac{(Mh)^2}{2} & 0 & & \\ 0 & 0 & \cdots & 0 \end{vmatrix}, \tag{41}$$

$$v = \begin{vmatrix} 1^1 \\ 1^2 \\ \vdots \\ \vdots \\ \vdots \\ 1^M \end{vmatrix}, \tag{42}$$

Then for an N-th order polynominal fit (40) becomes

$$\delta x = v\delta x_M - C(\delta\hat{x}_0)_2 \tag{43}*$$

where:

$$C = [(C^{(2)}v) \quad (C^{(3)}v) \quad \cdots \quad (C^{(N)}v)] \tag{44}$$

$$C^{(N)} = \begin{vmatrix} \frac{h^n}{n!} & \frac{(2h)^n}{n!} & \cdots & \frac{(Mh)^n}{n!} \\ \frac{(2h)^n}{n!} & \vdots & & 0 \\ \vdots & \frac{(Mh)^n}{n!} & & \\ \frac{(Mh)^n}{n!} & 0 & & 0 \\ 0 & 0 & & 0 \end{vmatrix}, \tag{45}$$

In order to obtain an explicit expression for $(\delta\hat{x}_0)_2$ in this modified system, (43) is substituted in (34) to get

$$A_2^T[v\delta x_M - C(\delta\hat{x}_0)_2] = A_2^T A_2 (\delta\hat{x}_0)_2 \tag{46}$$

Solving (46) for $(\delta\hat{x}_0)_2$ and substituting the result into (36) gives the QD unconstrained parameter estimates $(\hat{x}_0)_2$:

$$(\hat{x}_0)_2 = (\bar{x}_0)_2 + [A_2^T(C+A_2)]^{-1}A_2^T v\delta x_M. \tag{47}$$

Using (47) in (14) gives the derivatives' estimates for any point within the span:

$$\hat{x}_m = \Phi^m\hat{x}_0 = \Phi^m \begin{vmatrix} (\hat{x}_0)_1 \\ (\hat{x}_0)_2 \end{vmatrix} = \Phi^m \begin{vmatrix} \bar{x}_0 \\ \bar{x}_0 \\ (\bar{x}_0)_2 + [A_2^T(C+A_2)]^{-1}A_2^T v\delta x_M \end{vmatrix}$$

$$= \Phi^m\bar{x}_0 + \Phi_2^m [A_2^T(C+A_2)]^{-1}A_2^T v\delta x_M \tag{48}$$

where:

$$\Phi_2^m \equiv \begin{vmatrix} \frac{(mh)^2}{2} & \frac{(mh)^3}{3!} & \cdots & \frac{(mh)^N}{N!} \\ mh & \frac{(mh)^2}{2} & & \cdot \\ 1 & mh & & \cdot \\ 0 & 1 & & \cdot \\ \cdot & 0 & & \\ \cdot & \cdot & & \\ \cdot & \cdot & & mh \\ & \cdot & & \\ 0 & 0 & & 1 \end{vmatrix} \tag{49}$$

If we set m=M we have the desired QD matrix-vector formulation

$$\hat{x}_M = \bar{x}_M + w\delta x_M, \tag{50}*$$

where:

$$\bar{x}_M = \Phi\hat{x}_M = \Phi^M\bar{x}_0 = \Phi^{M+1}\hat{\dot{x}}_0 \tag{51}*$$

$$w = \Phi_2^M\,[A_2^T(C+\Lambda_2)]^{-1}\,\Lambda_2^T v. \tag{52}*$$

$\hat{\dot{x}}_M$ = real-time derivative estimates obtained in preceding computing step.

$$\delta x_M = x_M - \bar{x}_M \tag{53}*$$

In the QD filter the intercept and slope constraints are applied by (52) in which Φ_2^M effects only the corrections on the unconstrained parameters $(\hat{x}_0)_2$ to the real-time derivative estimates. (51) is written in several alternate forms to show that the parameter estimates $\hat{x}_0$ evaluated at m=0 in the preceding step are implicitly contained in the QD formulation.

The smoothed QD outputs (i.e., those nearest the point of constraint) are given by (14):

$$\hat{x}_0 = \Phi^{-M}\hat{x}_M \tag{54}*$$

so that the smoothed QD outputs are precisely the parameter estimates at the point of constraint.

Extensions and Applications of the QD Filter.

It is a very simple matter to mechanize the QD filter with a "variable span" which is dynamically self-adjusting (i.e.,adaptive) to specified properties of the current data. At WSMR, for example, the QD filter in the flight safety program for PERSHING overflight support mentioned earlier uses about a three-to-one variation in span length which automatically adjusts to the current magnitude of the total acceleration, i.e.,the greater the acceleration, the shorter the span. This feature effects a drastic reduction in velocity overshoot because the span is at its minimum value during the "staging" of this event. Although a little extra storage is required for the precomputed coefficients for each span length, the program logic for this feature is still quite simple.

It should be pointed out that variable span length can be adapted to almost any type of filter. The adaptation to QD happens to be extremely simple.

Reference 1 includes a description of the DFX digital filter developed by the writer in the Summer of 1964. Except for the manner in which the corrections for the unconstrained parameters are computed, QD and DFX employ the same basic recursive philosophy. In the Spring of 1965 the writer presented an informal paper to the Data Reduction and Computer Working Group of the Inter-Range Instrumentation Group at the Eastern Test Range (AFETR). In addition to explaining the basic DFX theory, the paper described an extremely effective technique for inserting independently measured derivative information into the filter performing its normal filtering functions. More specifically, it was pointed out that measured velocity data could be simultaneously inserted into a DFX filter accepting radar position data to essentially eliminate such errors as "overshoot" at motor burnout or other events generating abrupt changes in acceleration. To do this, the QD or DFX predicted velocity estimate $\bar{\dot{x}}_M$ is replaced directly with the corresponding measured velocity component v so that the predicted position estimate in a second order filter,

$$\bar{x}_M = \hat{\bar{x}}_M + \frac{h}{2}(\bar{\dot{x}}_M + \hat{\bar{\dot{x}}}_M), \tag{55}$$

would be replaced by

$$\bar{x}_M = \hat{\bar{x}}_M + \frac{h}{2}(v + \hat{\bar{\dot{x}}}_M). \tag{56}$$

It may be argued that this procedure does not buy anything because the velocity data are already available. This is an erroneous conclusion because, with inserted velocity data, the filter span can be significantly increased to provide dramatic improvement in filtering the radar data with no increased distortion of the true information in the radar data. In effect, then, the filter is operating on the difference between the trajectory described by the radar data and the trajectory described by the integration of the velocity data. This unique capability has not been used in practice at WSMR because suitable measured velocity data have not been available. The writer, however, has tested this technique in a wide variety of simulated situations with convincing results. This same improvement in performance has also been noted when velocity data along with position data have been inserted into the Kalman filter.

At the AFETR meeting in the Spring of 1965, the writer also described how this technique could be extended to the insertion of acceleration data. In filtering missile trajectory data, for example, measured acceleration data α_M obtained via telemetry can be inserted into the filter along with the measured position data by replacing the predicted velocity formula

$$\bar{\dot{x}}_M = \hat{\dot{x}}_M + h\hat{\ddot{x}}_M \tag{57}$$

with

$$\bar{\dot{x}}_M = \hat{\dot{x}}_M + h\alpha_M. \tag{58}$$

A special case of this acceleration data insertion has been used very successfully at WSMR for more than two years in the flight-safety support program mentioned earlier. In this case, a burnout signal generated from telemetered data is made available to the real-time computer program. When this burnout signal arrives the $\bar{\dot{x}}_M$ components of the QD filters are replaced by computed gravity components for the next several computing cycles. In other words, when the burnout event occurs, the QD filters are forced to forget about acceleration history. This procedure essentially eliminates the instantaneous impact prediction overshoot problem. This procedure also suggests that much more effective filtering of radar data can be obtained from a ballistic missile with negligible drag by inserting the gravity components into the QD filters.

It should be pointed out here that, when velocity or acceleration data are inserted into a QD filter, the smooth data formula (54) is no longer valid because the inserted data are arbitrary as far as the filter is concerned. (54) is based on a constant N-th order derivative across the filter span, which condition is obviously violated with the insertion of arbitrary derivative data.

We have also experimented extensively with QD filter arrangements using a simple deterministic mathematical model to generate the inserted acceleration data. As in the Kalman filter, the effectiveness of this arrangement is highly dependent on how well the trajectory generated by the model matches the measured trajectory. Feedback techniques have been used to force the measured trajectory data to dynamically update (correct) the model coefficients. Including a model with the QD filter can provide performance comparable to the Kalman filter in relatively simple applications, e.g., an N-station radar data reduction. When the application becomes complex, e.g., the dozens of coefficients contained

in a 6-degree of freedom model with both radar data and multichannel telemetry data inputs, the systematic Kalman formulation may be the only way to arrive at a near optimum solution. In real-time applications, on the other hand, the computing efficiency of the QD filter with inserted derivative data will provide performance comparable to the Kalman filter,which could not be used at all because of its excessive computing time.

From the foregoing discussion it is clear that the technique of inserting derivative data into a QD filter along with the normal input data and the comparison with Kalman performance is a new subject worthy of extensive investigation and is certainly beyond the scope of this report.

APPENDIX I.

$$\hat{x}_m = \hat{x}_0 + mh\dot{\hat{x}}_0 + \frac{(mh)^2}{2}\ddot{\hat{x}}_0 + \cdots + \frac{(mh)^N}{N!}\hat{x}_0^{(N)}, \quad 0 \leq m \leq M \tag{1}$$

[4] $$\hat{x}_i = \sum_{n=0}^{N} \frac{(ih)^n}{n!}\hat{x}_0^{(n)}, \quad i = 1, 2, \ldots M, \; N<M.$$

* * * * *

$$\psi = \sum_{m=0}^{M} \varepsilon_m^2 = \sum_{m=0}^{M} (x_m - \hat{x}_m)^2 \tag{2}$$

[3] $$\sum_{m=1}^{M} (x_m - \hat{x}_m)^2 = \text{Minimum.}$$

* * * * *

$$x - e = \hat{x} = A\hat{x}_0 \tag{3}$$

[4] $$\hat{x}_i = \sum_{n=0}^{N} \frac{(ih)^n}{n!}\hat{x}_0^{(n)}, \quad i = 1, 2, \ldots M, \; N<M.$$

* * * * *

$$\psi = e^T e = (x-\hat{x})^T(x-\hat{x}) \tag{4}$$

[3] $$\sum_{m=0}^{M} (x_m - \hat{x}_m)^2 = \text{Minimum}$$

* * * * *

$$\left.\frac{\partial \psi}{\partial x_0}\right|_{x_0=\hat{x}_0} = -2(x-\hat{x})^T A = -2A^T(x-\hat{x}) = 0 \tag{10}$$

[5] $$\sum_{m=1}^{M} m^j (x_m - \hat{x}_m) = 0, \quad j = 0, 1, \ldots N.$$

* * * * *

$$\hat{x}_0 = [A^T A]^{-1} A^T x, \tag{11}$$

[9] $$\hat{x}_0^{(j)} = \sum_{k=0}^{N} \bar{a}_{jk} S_k, \quad j = 0, 1, \ldots N.$$

$$[\bar{a}_{jn}] = [a_{jn}]^{-1}$$

[7] $$a_{jn} = \frac{h^n}{n!} \sum_{m=1}^{M} m^{j+n}$$

[8] $$S_j = \sum_{m=1}^{M} m^j x_m$$

* * * * *

$$\hat{x} = A[A^T A]^{-1} A^T x. \tag{12}$$

[11] $$\hat{x}_i^{(j)} = \sum_{k=0}^{N} b_{ik}^{(j)} S_k, \quad j = 0; \quad i = 1, 2, \ldots M$$

[12] $$b_{ik}^{(j)} = \sum_{n=j}^{N} \frac{(ih)^{n-j}}{(n-j)!} \bar{a}_{nk}.$$

* * * * *

$$\begin{aligned}
\hat{x}_m &= \hat{x}_0 + mh\hat{\dot{x}}_0 + \frac{(mh)^2}{2} \hat{\ddot{x}}_0 + \ldots + \frac{(mh)^N}{N!} \hat{x}_0^{(N)} \\
\hat{\dot{x}}_m &= \hat{\dot{x}}_0 + mh\hat{\ddot{x}}_0 + \ldots + \frac{(mh)^{N-1}}{(N-1)!} \hat{x}_0^{(N)} \\
&\vdots \\
\hat{x}_m^{(N)} &= \hat{x}_0^{(N)}
\end{aligned} \tag{13}$$

$$\hat{x}_m = \Phi^m \hat{x}_0 \tag{14}$$

[10] $$\hat{x}_i^{(j)} = \sum_{n=j}^{N} \frac{(ih)^{n-j}}{(n-j)!} \hat{x}_0^{(n)}, \quad j = 0, 1, \ldots N.$$

* * * * *

$$\hat{x}_m = \Phi^m [A^T A]^{-1} A^T x \tag{17}$$

[11] $$\hat{x}_i^{(j)} = \sum_{k=0}^{N} b_{ik}^{(j)} S_k, \quad j = 0, 1, \ldots N, \; i = 1, 2, \ldots M$$

[12] $$b_{ik}^{(j)} = \sum_{n=j}^{N} \frac{(ih)^{n-j}}{(n-j)!} \bar{a}_{nk}$$

* * * * *

$$\frac{\partial \psi}{\partial (x_0)_2} = -2(x_c - \hat{x}_c)^T A_2 = -2A_2^T (x_c - \hat{x}_c) = 0 \tag{28}$$

[15] $$\sum_{m=1}^{M} m^j (x_m - \hat{x}_m) = 0, \quad j = 2, 3, \ldots N.$$

* * * * *

$$(\hat{x}_0)_2 = [A_2^T \Lambda_2]^{-1} A_2^T x_c \tag{29}$$

[19] $$\hat{x}_0^{(j)} = \sum_{k=2}^{N} \bar{a}_{jk} \bar{S}_k, \quad j = 2, 3, \ldots N.$$

[18] $$\bar{S}_j = \sum_{m=1}^{M} m^j - \hat{x}_0 \sum_{m=1}^{M} m^j - h\hat{\dot{x}}_0 \sum_{m=1}^{M} m^{j+1}$$

* * * * *

$$\hat{x}_m = \Phi^m \hat{x}_0 = \Phi^m \begin{vmatrix} (\hat{x}_0)_1 \\ (\hat{x}_0)_2 \end{vmatrix} = \begin{vmatrix} \hat{x}_0 = \bar{x}_0 \\ \hat{\dot{x}}_0 = \bar{\dot{x}}_0 \\ [A_2^T A_2]^{-1} A_2^T x_c \end{vmatrix} \tag{30}$$

[20] $$\hat{x}_i = \hat{x}_0 + ih\hat{\dot{x}}_0 + \sum_{n=2}^{N} \frac{(ih)^n}{n!} \hat{x}_0^{(n)}$$

[21] $$\hat{\dot{x}}_i = \hat{\dot{x}}_0 + \sum_{n=2}^{N} \frac{(ih)^{n-1}}{(n-1)!} \hat{x}_0^{(n)}$$

[22] $$\hat{x}_i^{(j)} = \sum_{n=j}^{N} \frac{(ih)^{n-j}}{(n-j)!} \hat{x}_0^{(n)}, \quad j = 2, 3, \ldots N$$

* * * * *

$$A_2^T x_c = A_2^T A_2 (\hat{x}_0)_2 \tag{31}$$

[17] $$\sum_{n=2}^{N} a_{jn} \hat{x}_0^{(n)} = \bar{s}_j, \quad j = 2, 3, \ldots N$$

* * * * *

$$\delta x = x - A_1 (\hat{x}_0)_1 - A_2 (\bar{x}_0)_2 \tag{33}$$

[38] $$\overline{\Delta x}_m = x_m - \bar{x}_m$$

* * * * *

$$A_2^T \delta x = A_2^T A_2 \delta \hat{x}_0 \tag{34}$$

[37] $$\sum_{m=1}^{M} m^j \overline{\Delta x}_m = \sum_{n=2}^{N} a_{jn} \delta x_0^{(n)}, \quad j = 2, 3, \ldots N$$

* * * * *

$$\delta\hat{x}_0 = (\hat{x}_0)_2 - (\bar{x}_0)_2 \tag{35}$$

[34] $$\delta x_0^{(j)} = \hat{x}_0^{(j)} - \bar{x}_0^{(j)} \quad j = 2, 3, \ldots N$$

* * * * *

$$\delta\hat{x}_0 = [A_2^T A_2]^{-1} A_2^T \delta x \tag{37}$$

[40] $$\delta x_0^{(j)} = \sum_{n=2}^{N} \bar{a}_{jn} \bar{\bar{s}}_n$$

[39] $$\bar{\bar{s}}_j = \sum_{m=1}^{M} m^j \overline{\Delta x}_m, \quad j = 2, 3, \ldots N$$

* * * * *

$$(\hat{x}_0)_2 = (\bar{x}_0)_2 + [A_2^T A_2]^{-1} A_2^T \delta x \tag{38}$$

[46] $$\hat{x}_M^{(j)} = \bar{x}_M^{(j)} + \sum_{k=2}^{N} C_{Mk}^{(j)} \bar{\bar{s}}_k, \quad j = 0, 1, \ldots N$$

[47] $$C_{Mk}^{(j)} = \sum_{n=2}^{N} \frac{(Mh)^{n-j}}{(n-j)!} \bar{a}_{kn} \quad j = 0, 1$$

[48] $$C_{Mk}^{(j)} = \sum_{n=j}^{N} \frac{(Mh)^{n-j}}{(n-j)!} \bar{a}_{kn} \quad j = 2, 3, \ldots N$$

* * * * *

$$\delta x = v\delta x_M - C(\delta\hat{x}_0) \tag{43}$$

[60] $$\Delta x_i = \overline{\Delta x}_M - \sum_{p=i}^{M} \sum_{n=2}^{N} \frac{(ph)^n}{n!} \delta x_{0p}^{(n)}$$

* * * * *

$$\hat{x}_M = \bar{x}_M + w\delta x_M \tag{50}$$

[68] $$\hat{x}_M^{(j)} = \bar{x}_M^{(j)} + \bar{Q}_j \overline{\Delta x}_M, \quad j = 0, 1, \ldots N$$

[69] $$\bar{Q}_j = \sum_{n=p}^{N} \frac{(Mh)^{n-j}}{(n-j)!} Q_n, \quad \begin{matrix} p = 2 \text{ for } j = 0, 1 \\ p = j \text{ for } j = 2, 3, \ldots N \end{matrix}$$

[66] $$Q_j = \sum_{n=2}^{N} \bar{q}_{jn} \sum_{m=1}^{M} m^n, \quad j = 2, 3, \ldots N$$

$$[\bar{q}_{jn}] = [q_{nj}]^{-1}$$

[64] $$q_{jn} = \frac{h^n}{n!} \sum_{m=1}^{M} m^j \sum_{p=m}^{M} p^n$$

* * * * *

$$\bar{x}_M = \bar{\Phi}\hat{x}_M = \Phi^M \bar{x}_0 = \Phi^{M+1}\hat{x}_0 \tag{51}$$

[30] $$\bar{x}_M^{(j)} = \sum_{n=j}^{N} \frac{h^{n-j}}{(n-j)!} \bar{x}_{M-1}^{(n)}, \quad j = 0, 1, \ldots N$$

* * * * *

$$w = \Phi_2^M [A_2^T(C+A_2)]^{-1} A_2^T v \tag{52}$$

[69] $$\bar{Q}_j = \sum_{n=p}^{N} \frac{(Mh)^{n-j}}{(n-j)!} Q_n, \quad \begin{matrix} p = 2 \text{ for } j = 0, 1 \\ p = j \text{ for } j = 2, 3, \ldots N \end{matrix}$$

[66], [64] above

* * * * *

$$\delta x_M = x_M - \overline{x}_M \tag{53}$$

[41] $$\overline{\Delta x_M} = x_M - x_M$$

* * * * *

$$\hat{x}_0 = \Phi^{-M}\hat{x}_M \tag{54}$$

[51] $$\hat{x}_1^{(j)} = \sum_{n=j}^{N} \frac{(h-Mh)^{n-j}}{(n-j)!}\hat{x}_M^{(n)}, \quad j = 0, 1, \ldots N.$$

[51a] $$\hat{x}_0^{(j)} = \sum_{n=j}^{N} \frac{(-Mh)^{n-j}}{(n-j)!}\hat{x}_M^{(n)}, \quad j = 0, 1, \ldots N$$

* * * * *

References

1. "QD - A New Efficient Digital Filter", W. A. McCool, Internal Memorandum No. 60, Aug. 1967, Analysis and Computation Directorate, White Sands Missile Range, New Mexico.

2. "Optimizing the Dynamic Parameters of a Track-While-Scan System", Jack Sklansky, RCA Review, June 1957, RCA Laboratories, Princeton, N.J.

3. "Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Equations", T.R. Bendict and G. W. Bordner, IRE Transactions on Automatic Control, 1962.

4. "Development of a Linear Recursive Filter", J. J. Lynn, Oct. 1964, In-Flight Analysis, Mathematical Services, RCA Service Co.